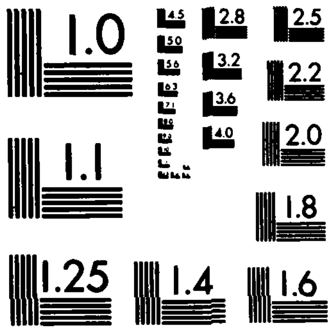
AD-A136 122   SEARCH WITH LIMITED RESOURCES(U) DUKE UNIV DURHAM NC   1/1
DEPT OF COMPUTER SCIENCE D C MUTCHLER FEB 83 CS11983-1
AFOSR-TR-83-1154 AFOSR-81-0221

UNCLASSIFIED   F/G 12/1   NL

END
FILMED
1-84
DTIC


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

AFOSR-TR- 83 - 1154

2

AD-A136122

SEARCH WITH LIMITED RESOURCES

by

David C. Mutchler

Department of Computer Science

Duke University

CS-1983-1

This research has been sponsored by the Air Force Office of Scientific Research, Air Force Systems Command under Grant AFOSR 81-0221.

8

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-1154	2. GOVT ACCESSION NO. AD-A136122	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SEARCH WITH LIMITED RESOURCES		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER CS-1983-1
7. AUTHOR(s) David C. Mutchler		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0221
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Duke University Durham NC 27706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research /NM Bolling AFB DC 20332		12. REPORT DATE FEB 83
		13. NUMBER OF PAGES 55
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Extended abstract submitted to 21st Southeast Region Conference of the ACM. Revised version to be submitted to AAAI Conference in Washington DC.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Search strategies; search methodlogy; evaluation function; decision trees; game trees.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Most game-playing programs make each move after conducting only a partial search of the game tree and applying a static evaluation function at the terminal nodes of that partial search. Given limited resources, what is the optimal partial search to perform? This report presents a model for investigating this question. Results (including the answer to the above question) are obtained for a restricted case of the model.

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

This research was spawned by a suggestion made by my advisor Don Loveland that I investigate an idea proposed by Tom Truscott called "minimum variance search". Many of the ideas herein were developed during meetings with Don, Tom, and Alan Biermann. Comments from these friends were a frequent source of inspiration.

Conversations with Bruce Ballard were also enlightening. My thanks go to him especially for supplying the key ideas of the proof of Result 5 in Section 3.3.3.

ABSTRACT

Most game-playing programs make each move after conducting only a partial search of the game tree and applying a static evaluation function at the terminal nodes of that partial search. Given limited resources, what is the optimal partial search to perform? This report presents a model for investigating this question. Results (including the answer to the above question) are obtained for a restricted case of the model.

1. Introduction

Scarcity is the cornerstone of economic theory. With chip technology growing by leaps and bounds, some deny the relevance of scarcity to computer problem solving. "Just buy a bigger machine," they say. Yet some search tasks, like the travelling salesman problem, chess-playing and optimal circuit design, have decision trees so huge as to preclude complete search in any current technology. This necessitates partial search, implemented by applying a heuristic evaluation function to certain interior nodes of the decision tree. The job of the evaluation function is to provide information about the subtrees below the evaluated node.

This report is an investigation of the search strategy rather than the move strategy, that is, which nodes should be evaluated as opposed to what decision should be made. A simple search strategy is uniform depth 1 search: evaluate each possible move directly, and select that which appears best. Such a strategy is like a child to whom "tomorrow" is synonymous with all the future. As the child grows to understand "next week" and "three years hence", so might our strategy expand to uniform depth 2 search, depth 3 search, and so on.

The "combinatorial explosion" [NILS 80] of most search problems keeps the search depth embarrassingly small. Fear of the "horizon effect" [TRUS 82], wherein important information lies just beyond the deepest level searched, encourages abandonment of uniform search. Instead, certain "promising" lines are searched deeply, while other possibilities are left unexplored. Such search is patterned after "insight", a typically human thought process.

These search strategies and others have long been known to the game-playing community [BIER 78]. The question of which strategy is best, however, remains unanswered. This report presents a model for investigating where in the decision tree limited search resources should be expended in order to aid decision-making. Analytic results are obtained for a restricted case.

2. The model

2.1. What every good model should be able to do

We wish to study the situation wherein limited resources prevent complete search to the end of the decision process. Perforce, uncertainty as to the correctness of our choices enters the stage. Uncertainty arrives in decision-making from many sources, including unpredictable opponents, other external events, errors in the evaluation function, and incomplete search. Our model must isolate this last source of uncertainty.

Our model must also make clear the distinction between a search strategy and a move strategy. The following example was suggested by conversations with Tom Truscott about minimum variance search [TRUS 79].

Prudence, appearing on "Let's Make A Deal", must choose either what is in the box beside her or what is behind the curtain on the stage. She knows (because she bribed a trusted stagehand before the show) that the box contains $2000 in cash. Careful records of the last forty-six episodes of the show indicate that the average curtain-item has value $1417, although some items behind the curtain have been worth as much as $5000. Clearly, her *move* strategy, based on current information, must be to take the box. But her *search* strategy should be to further explore the curtain, for example by signalling her thirteen-year-old cousin Ozzy to sneak behind it. The model herein must define precisely the notion of strategy, with both its components. Further, it should allow quantitative comparison of strategies.

2.2. Playing games

We model a decision process by a one-player game played on a game tree. The game tree may have any shape desired; the shape is known to the player. Each of the leaves (called **goal nodes**) of the game tree has a value, which may be any real number. The player begins at the root and makes sequential, irreversible moves until he reaches a goal node. The value of that node is the score the player achieves. The player is trying to maximize his expected score.

It remains to declare what information the player may acquire from search of the game tree. First we enrich our vocabulary.

Definition: To move **randomly** at a node means herein that the probability the player moves to one child is the same as the probability he moves to any other child of that node. Suppose a player moves randomly from node w until the end of the game. Then the score the player will achieve is a random variable X_w called the **blind search value of node w**. Its probability mass function (pmf) p_w is called the **blind search pmf of node w** and its expectation $E[X_w]$ is called the **blind search expectation of node w**.

Definition: To **explore** node w in the game tree means to be told the blind search pmf p_w of node w.

At any time during the game, the player may explore any node in the tree. This exploration models the generation of accurate, consistent, yet incomplete information from search. Further, the exploration is generic in nature, being tied to no particular game.

To model limited resources, we associate a cost of \$1 with each exploration of a node. The player begins with a limited but known budget. Generously, we give the player the blind search pmf of the root of the game tree free of charge.

2.3. Kibitzing

Definition: A **search strategy** is a collection of rules that specify which nodes of the game tree are to be explored at what times. A **move strategy** is a collection of rules that specify what moves to make during the game. A **strategy** is the union of a search strategy and a move strategy.

The search and move strategies may be dynamic and may communicate. That is, the result of one exploration or move may partially determine the succeeding portion of the search and move strategies. The strategies may also be probabilistic. The player is assumed to have a true random-number generator at his disposal.

Let us review the current status of our model. Someone (say Zeus) has supplied three game parameters:

1. the shape of the game tree,
2. the blind search pmf of the root of the tree, and
3. the player's budget.

The player knows the values of these three parameters. He brings with him a strategy, as defined above.

At this point the player could play the game, achieving some fated result. But as many gamblers have found to their dismay, there is a difference between the winning play and the correct one (obtained by playing the odds). We are interested in evaluating the average, rather than actual, score yielded by the player's strategy. To police and evaluate a strategy, we introduce a **meta-player (MP)**.

The MP simulates the player's strategy on each possible game tree. Recall that many game trees are possible from the player's viewpoint, because although the player knows the blind search pmf of the root, he does not know which goal nodes have which goal values. The cards have been dealt but not yet seen. For any assignment of goal values to goal nodes consistent with the given root blind search pmf, the MP simulates the player's strategy to find an average (expected) score. This score is an average rather than a single number because of possible probabilistic aspects of the player's strategy. The average is weighted according to the probability distribution specified by the player in his strategy.

Once these average scores are computed, the MP averages them over all possible game trees. How to weight this average has not yet been specified. It is reasonable, given the lack of additional information, to assume that all such game trees are equally likely. For emphasis, here is a precise statement of this assumption.

Meta-player assumption: Let a given game tree have shape Φ, and let its root have pmf p_r. Label its goal nodes $g = (g_1, g_2, \cdots, g_n)$. Let Γ be the set of all possible assignments of goal values $v = (v_1, v_2, \cdots, v_n)$ to the goal nodes, consistent with Φ and p_r. *We assume that all the elements of Γ are equally likely.*

2.4. A strategy unmasked

Enough of description! Watch the MP in action as he evaluates a strategy.

Zeus sends us a complete, binary, depth two tree. He says the root blind search pmf p_r is:

$$p_r(1) = p_r(0) = \frac{1}{2} .$$

The only trees consistent with these specifications are those with two goal nodes having value "1" and two having value "0". There are six such trees, pictured in Figure 1.

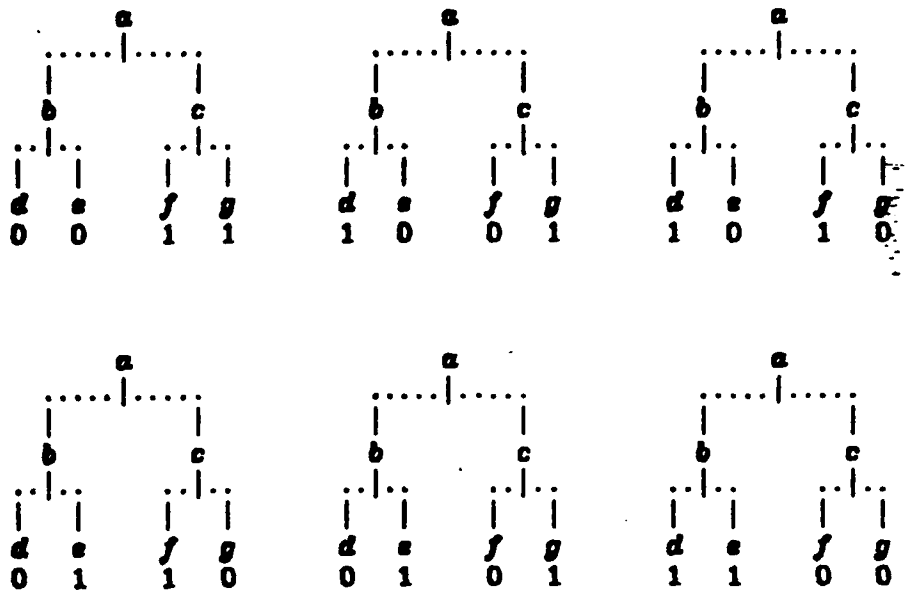


Figure 1.

The player Prometheus brings the following \$1 strategy.

1. He explores the left child b of the root (spending his \$1 there) and computes $E[X_b]$ from the information purchased. He computes $E[X_c] = 1 - E[X_b]$. Then Prometheus moves to the child with larger expectation; if the expectations are tied, he makes his first move randomly.

2. Prometheus makes his second move at random.

Note that Prometheus has used his god-given knowledge of root pmf p_a in his strategy.

To evaluate this strategy, the MP plays the game using the strategy on each of the six trees above. For example, on the first tree pictured, the search strategy discloses that $E[X_b] = 0$ and $E[X_c] = 1$, so Prometheus moves to node c. He then moves to either node f or node g (at random), scoring a "1" in either case. The MP records that Prometheus' strategy yields a score of "1" on tree 1.

Next the MP tries Prometheus' strategy on Tree 2 (the middle tree in the first row in Figure 1). This time

$$E[X_b] = E[X_c] = 0.5$$

The player makes both of his moves at random in Tree 2, receiving a score of either "1" or "0" but averaging a score of 0.5. The MP records that Prometheus' strategy yields an expected score of 0.5 on Tree 2. Note that this is an expected (average) score. In none of the six trees in Figure 1 can the actual score be 0.5.

Analysis of Trees 3 through 5 is symmetric to that of Tree 2. Analy of Tree 6 is symmetric to that of Tree 1. The meta-player assumption says that ix trees are weighted equally. The MP evaluation of this strategy is the average of evaluation on each tree, ie.,

$$(1.0 + 0.5 + 0.5 + 0.5 + 0.5 + 1.0) \div 6 = \frac{2}{3}$$

In no game does Prometheus actually receive a score of $\frac{2}{3}$. His strategy works better than this average on some trees and worse on others. When we compare two strategies, we will not claim that one strategy outperforms the other on all possible games, but only that one has higher average score than the other.

3. Restricting the Model to Limited Strategies on 0-1 Binary Trees

3.1. The Restrictions

In this section, we restrict the model in three ways.

1. All trees are finite complete binary trees.

2. "0" and "1" are the only two goal values allowed.

3. The only strategies allowed are those of the following form:

   a. The player moves randomly until he reaches node w at depth $k-1$ ($0 \leq k-1 <$ depth of tree).

   b. The player explores the two depth k children of node w. From the probability distributions received, he computes the blind search expectation of each of these children.

   c. The player moves to the child whose blind search expectation is the larger of the two.

   d. The player moves randomly for the rest of the game.

Call such a strategy the **one-time k strategy**.

3.2. Definitions

Note that in our model a finite complete binary 0-1 tree can be described by two parameters:

p = number of goal nodes with value "1", and

n = number of goal nodes.

This justifies the following definition.

Definition: A (p,n) tree is a complete binary tree with n goal nodes of which p have value "1" and the rest have value "0".

Note that the blind search expectation of a (p,n) tree is the average goal node value, ie., $\frac{p}{n}$.

The blind search expectation of a tree is the stick by which we measure the player's performance. It is the outcome we would expect the player to average if he were to play the game many times with no information to guide him. The question thus arises: *How much will information help the player?* The function in the next definition tells how much the player is helped by information, in this restricted version of the game.

Definition: The random variable $I(k,p,n)$ is the improvement for a (p,n) tree that a player gains over its blind search expectation by using the one-time k strategy. Let $f_k(p,n)$ denote its expected value $\mathbf{E}[I(k,p,n)]$.

The randomness in the random variable $I(k,p,n)$ arises from two sources: the randomness in the player's strategy and the random (p,n) tree selected. The former randomness obeys a uniform distribution when the player is using a one-time k strategy. The latter randomness obeys the uniform distribution given by the meta-player assumption, ie., that all (p,n) trees are equally likely.

$f_k(p,n)$ does not give an actual score the player will receive in any particular instance of a game played on a (p,n) tree. His strategy will work better in some instances and worse in others. $f_k(p,n)$ is an average over all possible games played on (p,n) trees.

3.3. Results and Conjectures

This section answers many interesting questions about the strategies in this restricted model. For each statement labelled "Result", its proof either follows the statement or is in Appendix A. The statements for which no explicit proof appears are labelled "Conjecture" and are supported by the computer results in Appendix B.

Unless otherwise stated, results about $f_k(p,n)$ are for integers k, p, n, with n a power of 2 ($n \geq 2$), $0 \leq p \leq n$, and $1 \leq k \leq \log n$. Throughout, $\binom{x}{y}$ denotes "x choose y" and is interpreted as 0 if $x < y$. Logarithms are base 2. The symbol ▪ denotes the end of a proof.

3.3.1. Computational Results

On the average, how much should the player expect to gain by doing a single exploration of two adjacent subtrees at level k of a (p,n) tree? That is, how does one compute $f_k(p,n)$? The results of this section give formulas for doing that computation. These results justify the programs written to compute $f_1(p,n)$ and $f_k(p,n)$ for various values of k, p and n.

Result 1: $f_k(p,n) = f_k(n-p,n)$.

Proof: Appendix A (section 5.1). ▪

This expresses the 0-1 symmetry. Recall that p and $n-p$ are the number of goal nodes with values "1" and "0" respectively. This result says that the improvement gained from a single exploration of two adjacent subtrees in a tree with p "1"s is exactly the same (on average) as that gained in a tree with p "0"s.

This result does not say that the average score achieved is the same in the two cases. It says only that the expected *increase* from the two (different) no-search scores is the same. Note that the result applies no matter when the one-time exploration is performed.

Most of the following formulas are true for all values of p, but the programs use Result 1 to halve the computational effort.

Result 2:

$$f_1(p,n) = \frac{2}{n\binom{n}{n/2}} \sum_{j=\lfloor p/2+1\rfloor}^{n/2} [2j-p]\binom{p}{j}\binom{n-p}{n/2-j}$$

$$= \frac{4}{n^2\binom{n}{n/2}} \left\lfloor \frac{p}{2}+1 \right\rfloor \left\lfloor \frac{n-p+1}{2} \right\rfloor \binom{p}{\lfloor p/2+1\rfloor}\binom{n-p}{\lfloor (n-p+1)/2\rfloor}$$

$$= \begin{cases} 0 & \text{if } p=0 \text{ or } p=n \\ \frac{1}{n} & \text{if } p=1 \text{ or } p=n-1 \\ \frac{n-p}{n-p+1} f_1(p-1,n) & \text{if } 2 \le p \le n-2 \text{ and } p \textit{ is even} \\ \frac{p}{p-1} f_1(p-1,n) & \text{if } 3 \le p \le n-3 \text{ and } p \textit{ is odd} \end{cases}$$

$$= \begin{cases} 0 & \text{if } p=0 \text{ or } p=n \\ \frac{1}{n} \prod_{j=1}^{\lfloor p/2\rfloor} \frac{n-2j}{n-2j+1} \prod_{j=1}^{\lfloor (p-1)/2\rfloor} \frac{2j+1}{2j} & \text{if } 1 \le p \le n-1 \end{cases}$$

Proof: Appendix A (section 5.2). ▪

These formulas give ways to compute $f_1(p,n)$. The next result will give $f_b(p,n)$ in terms of f_1.

The first formula in Result 2 is the naive computation of $f_1(p,n)$. Its proof is illustrative of the combinatorial meaning of $f_1(p,n)$. Exploring at the root of the tree means splitting the goal nodes into two halves and choosing the half with more "1"-valued goal nodes. In the first formula, the denominator $\binom{n}{n/2}$ is the number of ways to divide the n goal nodes into two halves. The sum is over the cases when j ones and $n/2-j$ zeros are in the selected half. The last two "choose" terms in the formula give the number of ways to do the split in case j. The improvement in each case is the $2j-p$ term inside the sum times the $\frac{1}{n}$ term outside the sum. The factor of 2 exists because of the symmetry of the cases and the fact that whichever half has more "1"-valued goal nodes is selected.

The first formula is a short sum but suffers from including terms with large integers. In particular, $\binom{n}{n/2}$ is too large for a PDP 11/70 double variable (64 bits) when $n \geq 256$ (depth 8 trees).

The second formula is a closed-form version of the first formula. It eliminates the short sum but suffers the same difficulty with large integers. Its proof consists only of combinatorial tricks applied to the first formula.

The third formula gives the recursive definition used by the programs to compute $f_1(p,n)$. The basis case is when p is 1; the recursion is on p. $f_1(p,n)$ is easily computed by this formula even for very large trees. Note the difference and symmetry of the odd/even cases. Result 6 states more clearly how $f_1(p,n)$ changes as a function of p. The proof of the third formula is by induction using the second formula. The difference in the odd/even cases occurs because the terms truncated in the second formula depend on whether p is odd or even.

The fourth formula in Result 2 is a non-recursive application of the third formula.

Result 3: for $k \geq 2$,

$$f_k(p,n) = \frac{1}{\binom{n}{m}} \sum_{j=0}^{m} \binom{p}{j} \binom{n-p}{m-j} f_1(j,m)$$

where $m = \frac{n}{2^{k-1}}$

Proof: In the one-time k strategy, the player explores two randomly selected adjacent subtrees, each with root at level k of the (p,n) tree, and moves to whichever subtree has more goal nodes with value "1". In other words, the player moves to a random node w at level $k-1$, and applies the one-time 1 strategy to the subtree with root w.

Divide the possible outcomes into cases: case j means the node w subtree has j goal nodes below it with value "1". Since node w is at level $k-1$, the expected improvement over blind search in case j is $f_1(j,m)$, where m = size of the node w subtree $= \frac{n}{2^{k-1}}$.

We must weight each case appropriately. The number of ways to select the j goal nodes with value "1" in the node w subtree is $\binom{p}{j}$. The number of ways to select the remaining goal nodes in the node w subtree is $\binom{n-p}{m-j}$. There are $\binom{n}{m}$ ways to select a subtree with root at level $k-1$, so the weight of case j is

$$\frac{\binom{p}{j}\binom{n-p}{m-j}}{\binom{n}{m}}$$

Multiplying the expected result for each case by the weight of that case and summing over all the cases yields Result 3. ▪

This formula expresses $f_k(p,n)$ in terms of f_1. For $p \leq n/2$, the non-zero terms of the sum are those for which j lies between 0 and min (m,p). For $p > n/2$, the non-zero terms are those for which j lies between max $(0, m - (n-p))$ and m. In either case, the formula is a short sum. Note that the sum is longest (m is largest) when $k = 2$.

The f_1 terms are easily computed from Result 2. However, the combinatorial terms in $f_k(p,n)$ may still be quite large. For example, f_2 requires $\binom{n}{n/2}$ which is too large for a PDP 11/70 double variable (64 bits) when $n \geq 256$ (depth 8 trees). The problem in calculating $f_k(p,n)$ lies not in the time required to compute the sum but rather in difficulties in representing quotients whose numerator and denominator are both quite large.

3.3.2. Special Cases

This proposition gives simple formulas for $f_1(p,n)$ and $f_k(p,n)$ in three special cases. The first two cases are cases for which all or all but one of the goal nodes are the same. The third special case is a nice formula for exploration at the bottom of the game tree.

Result 4: $f_k(0,n) = f_k(n,n) = 0$

$$f_k(1,n) = f_k(n-1,n) = \frac{1}{n}$$

$$f_{\log n}(p,n) = \frac{p\,(n-p)}{n\,(n-1)} \quad \text{for } n \geq 4$$

Proof: The first formula follows immediately from Result 2 and Result 3. The $\binom{p}{j}$ or $\binom{n-p}{m-j}$ term in Result 3 is always 0 if $p = 0$ or $p = n$, except for the $j = 0$ or $j = m$ case, for which $f_1(j,m) = 0$ by Result 2.

The second formula also follows from Result 2 and Result 3. If $p = 1$, the sum in Result 3 consists of two summands, the first of which is zero and the second of which is easily found from Result 2. The $p = n-1$ case can be computed similarly or by using the symmetry in Result 1.

The third formula follows from Result 3 and the other two formulas in this result. If $k = \log n$, the sum in Result 3 consists of three summands ($m = 2$). The f_1 term in each is computed from the first two formulas in Result 4. ∎

The first formula above says that trees whose goal values are all "0"s or all "1"s do not improve upon closer examination. Trees with all "0"s are hopeless and those with all "1"s are perfect already.

From the second formula we see that exploring trees with a single black sheep enables the player to double his expected score, which improves from $\frac{1}{n}$ to $\frac{2}{n}$. It seems surprising that the improvement is independent of the level k of exploration.

The last formula above gives the improvement gained by exploring at the last possible moment, that is, just before the player's last move. The formula is pleasantly simple. Result 7, Result 10 and Result 14 further explain the behavior of $f_{\log n}(p,n)$. Each is an easy corollary to Result 4.

Other special cases for f_k can be similarly computed from Result 3. However, as m in Result 3 increases linearly, the number of terms in the Result 3 sum increases exponentially. Application of Result 3 to increasingly large subtrees becomes computationally difficult quite quickly.

3.3.3. Which One-time k Strategy is Best?

The player is most interested in how he should play to maximize his game score. He wants the answer to the question: *Given a (p,n) tree, and given that the player can explore two adjacent subtrees exactly once in the game but must make all other moves blindly, when should the player do that exploration?*

The corollary to Result 5 answers this question. Result 5 is the stronger statement of monotonicity.

Result 5 [Ballard and Mutchler]: for $1 \le k < \log n$,

$$f_k(p,n) \le f_{k+1}(p,n)$$

with equality iff $p = 0$ or $p = 1$ or $p = n-1$ or $p = n$.

Further, the proof does not depend on the goal node values being restricted to "0" and "1", and works for any complete n-ary tree.

Corollary [Ballard and Mutchler]: for $1 \le k < \log n$,

$$f_k(p,n) \le f_{\log n}(p,n)$$

with equality iff $p = 0$ or $p = 1$ or $p = n-1$ or $p = n$.

Further, the proof does not depend on the goal node values being restricted to "0" and "1", and works for any complete n-ary tree.

Proof: The proof of Result 5 is in Appendix A (section 5.3). The key ideas in it were produced by Bruce Ballard following a conversation with the author. The corollary is an immediate consequence of the result. ▪

Result 5 is the strongest result in this report. It says that the player does better (on average) at each moment of the game to put off his single exploration until the next move, up to his last move. The corollary is immediate: the player should do his single exploration at the last possible moment, ie., just before the player makes his last move. This authoritatively answers the question of which of the strategies allowed in this restricted model is best.

This result does not say that for every tree, the player necessarily will do better to postpone his exploration. For example, consider the second through fifth trees of Figure 1 in Section 2.4. If the player explores before his first move and then makes his second move at random, he will see that both halves yield an average score of 0.5, which he will achieve (on average). In those four trees, if he makes his first move randomly and then explores the two goal nodes beneath him, he will always find a goal node with value "1", for an average score of 1.0. In those four trees, the player is better off postponing the exploration until prior to his second move.

But now look at the first and last trees of Figure 1 in Section 2.4. In those, if the player explores before his first move he will move toward the half with all "1"s, for an average score of 1.0. But if he postpones exploration until after his first move, half of the time he will find only "0"s below him, for an average score of 0.5. The player is better off in the first and last trees to explore before his first move.

Result 5 says that if we average each technique over all possible trees, assuming that each such tree is equally likely, then the player does better (on average) to do his exploration at the lower level of the tree.

Much of the strength of this result lies in it not depending on the values of the goal nodes being restricted to "0" and "1". The proof of the result involves no combinatorics. It uses only a fundamental property of the "maximum" function and the linearity of expected value.

This result suggests that a similar result might hold if the player is allowed to do more than one exploration. Further investigation of this extension will be forthcoming.

3.3.4. For What Trees Does Search Most Help?

The player is also interested in knowing for which trees search is most helpful. Result 4 says that search doesn't help at all if the goal nodes are all the same. Results 6 through 8 answer the question: *In what game trees should the player be most willing to pay for the right to explore? In terms of our model, what happens to $f_k(p,n)$ as p varies from 0 to n, while k and n are held fixed?*

Result 6:

$$f_1(p,n) > f_1(p-2,n) \quad \text{if } 2 \le p \le \frac{n}{2}$$

$$f_1(p,n) > f_1(p+2,n) \quad \text{if } \frac{n}{2} \le p \le n-2$$

$$f_1(p,n) < f_1(p-1,n) \quad \text{if } p \text{ is even}$$

$$f_1(p,n) > f_1(p-1,n) \quad \text{if } p \text{ is odd}$$

In particular, for fixed n, $f_1(p,n)$ is largest when $p = \frac{n}{2} - 1$ or $p = \frac{n}{2} + 1$.

Proof: Appendix A (section 5.4). ▪

This result states two things. First, if we look at only even p or at only odd p, then $f_1(p,n)$ increases as p increases until p reaches midway $(\frac{n}{2})$; then $f_1(p,n)$ decreases as p continues to increase to n. This relationship is illustrated for $n = 128$ in Figure 2.

The second relationship given by this result is that $f_1(p,n)$ is bigger for odd values of p than for either even neighbor. That is, for odd p,

$$f_1(p,n) > f_1(p-1,n) \quad \text{and} \quad f_1(p,n) > f_1(p+1,n)$$

This relationship is illustrated for $n = 128$ in Figure 3.

Suppose that prior to his first move the player will do a single exploration of the two subtrees beneath the root of the game tree. For such a player, Result 6 means that he expects to learn more in trees whose goal nodes are about half "1"s and half "0"s than in trees whose goal nodes are predominantly "1"s or predominantly "0"s.

The proofs of the statements is Result 6 use the third formula in Result 2. The proofs do not explain the odd/even difference. However, one would expect more

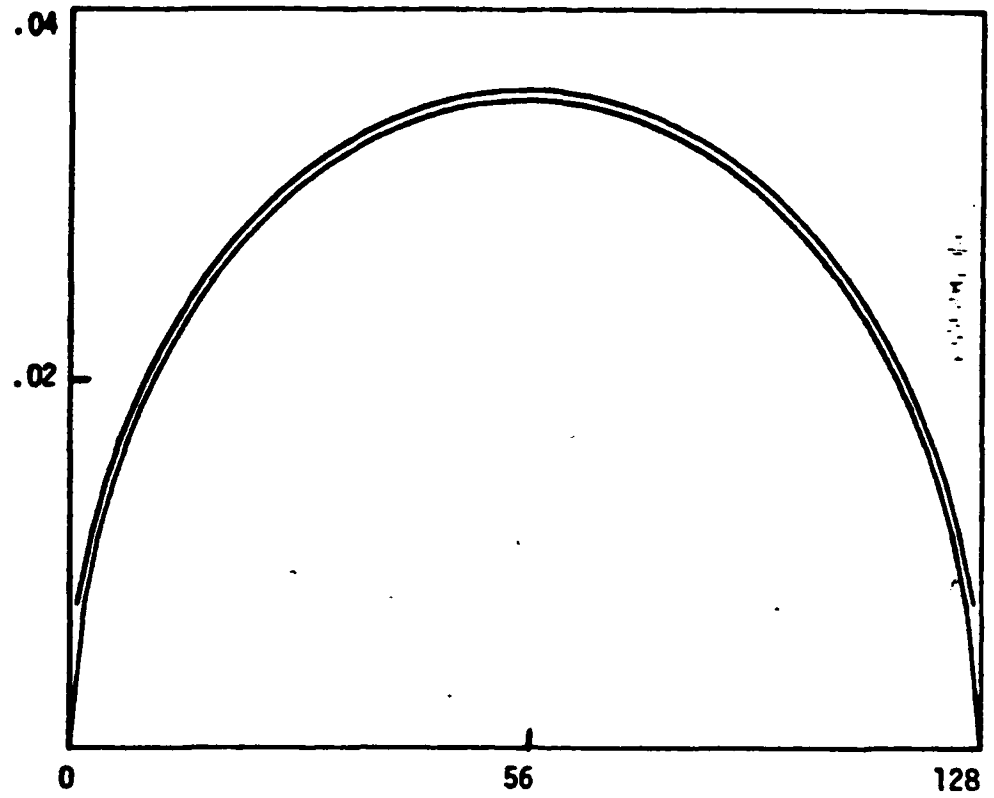


Figure 2: p *vs.* $f_1(p, 128)$, drawn as continuous curves. Top curve is for odd p. Bottom curve is for even p.

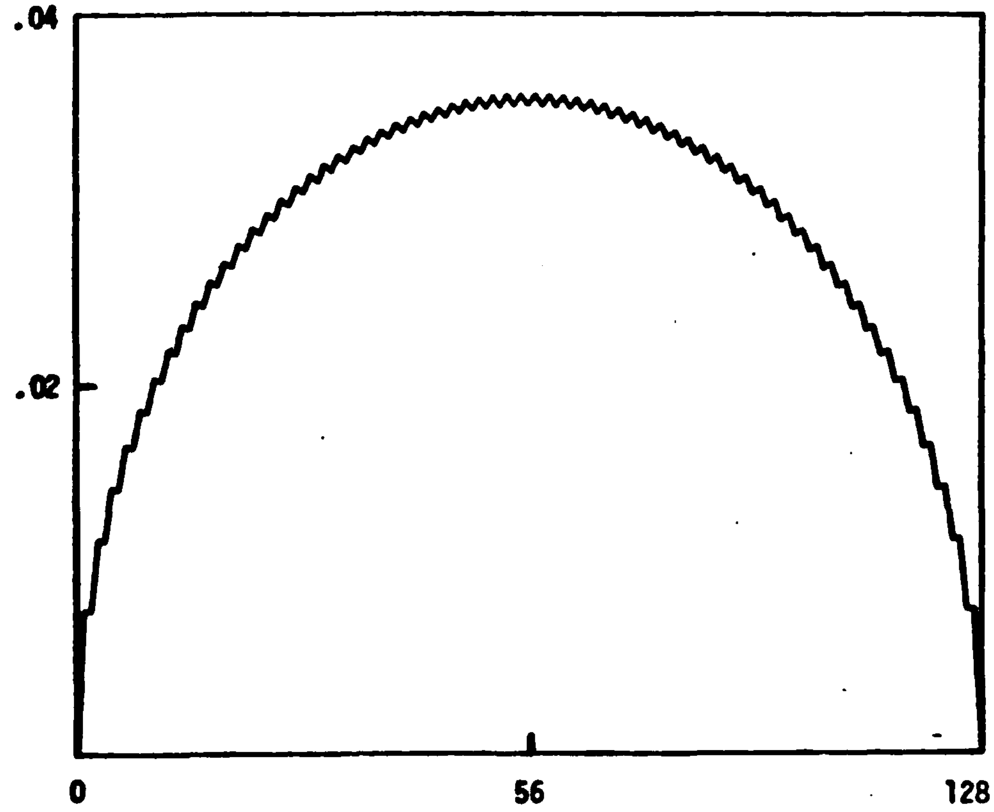


Figure 3: p *vs.* $f_1(p, 128)$, drawn as a continuous curve.

improvement in (p,n) trees with p odd than in those with p even, since only in the even case is it possible for the player to explore and nonetheless achieve no improvement in expected result. (This happens if the two halves of the tree each contain exactly half of the goal nodes with value "1".)

Result 7: for fixed $n \geq 4$,

- $f_{\log n}(p,n)$ is an increasing function of p for $0 \leq p \leq \frac{n}{2}$, decreasing for $\frac{n}{2} \leq p \leq n$, and convex throughout. In particular, for fixed $n \geq 4$, $f_{\log n}(p,n)$ is largest when $p = \frac{n}{2}$.

Proof: From Result 4,

$$f_{\log n}(p,n) = \frac{p\,(n-p)}{n\,(n-1)}$$

Viewing this as a continuous function of p, we see that

$$\frac{\partial}{\partial p} f_{\log n}(p,n) = \frac{n-2p}{n\,(n-1)} \begin{cases} >0 & \text{if } 0 \leq p < \frac{n}{2} \\ =0 & \text{if } p = \frac{n}{2} \\ <0 & \text{if } \frac{n}{2} < p \leq n \end{cases}$$

Also, $\frac{\partial^2}{\partial p^2} f_{\log n}(p,n) = -2 < 0$.

The result for integer p follows from the continuous case. ▪

Recall that $f_{\log n}(p,n)$ gives the expected improvement when our strategy is to perform our one-time exploration at the last possible moment. Result 7 and Result 6 show that behavior with respect to proportion of goal nodes having value "1" is the same for the player who spends his money at the last moment as for the player who spends his money as quickly as possible, except for the odd/even disparity.

The topmost curve in Figure 4 illustrates Result 7 for $n = 128$.

Conjecture 8: for $k \geq 2$,

$f_k(p,n)$ is an increasing function of p for $0 \leq p \leq \frac{n}{2}$, decreasing for $\frac{n}{2} \leq p \leq n$, and convex throughout. In particular, for fixed n and fixed $k \geq 2$, $f_k(p,n)$ is largest when $p = \frac{n}{2}$.

This conjecture says that (except for $f_1(p,n)$) the graph of $f_k(p,n)$ as a function of p is convex and symmetric about $\frac{n}{2}$. Computer results in Appendix B verify the conjecture for $n \leq 128$. The results for $n = 128$ are graphed in Figure 4. It seems highly likely that the conjecture is in fact true for larger values of n as well, ie., that all one-time k strategies behave similarly with respect to the make-up of the goal values.

3.3.5. What Happens in Large Trees?

Most game trees are very large. This section examines the behavior of $f_k(p,n)$ in large trees. The results answer the following questions. *Suppose that we double the number of goal nodes (ie., add a level to the tree) while maintaining the same proportion of goal nodes with value "1". How does this change $f_k(p,n)$? Does the behavior converge as the number of goal nodes gets large?*

Result 9: $f_1(2p,2n) \leq f_1(p,n)$ with equality iff $p=0$ or $p=n$.

Proof: Appendix A (section 5.5). ▪

This result says that adding a level to the tree while maintaining the same proportion of goal nodes with value "1" causes the player to gain less (on average) from search, provided that the search is a single exploration of the two subtrees of the root prior to his first move. The result is believable: in the expanded tree the player receives his information one step further from the goal nodes than he does in

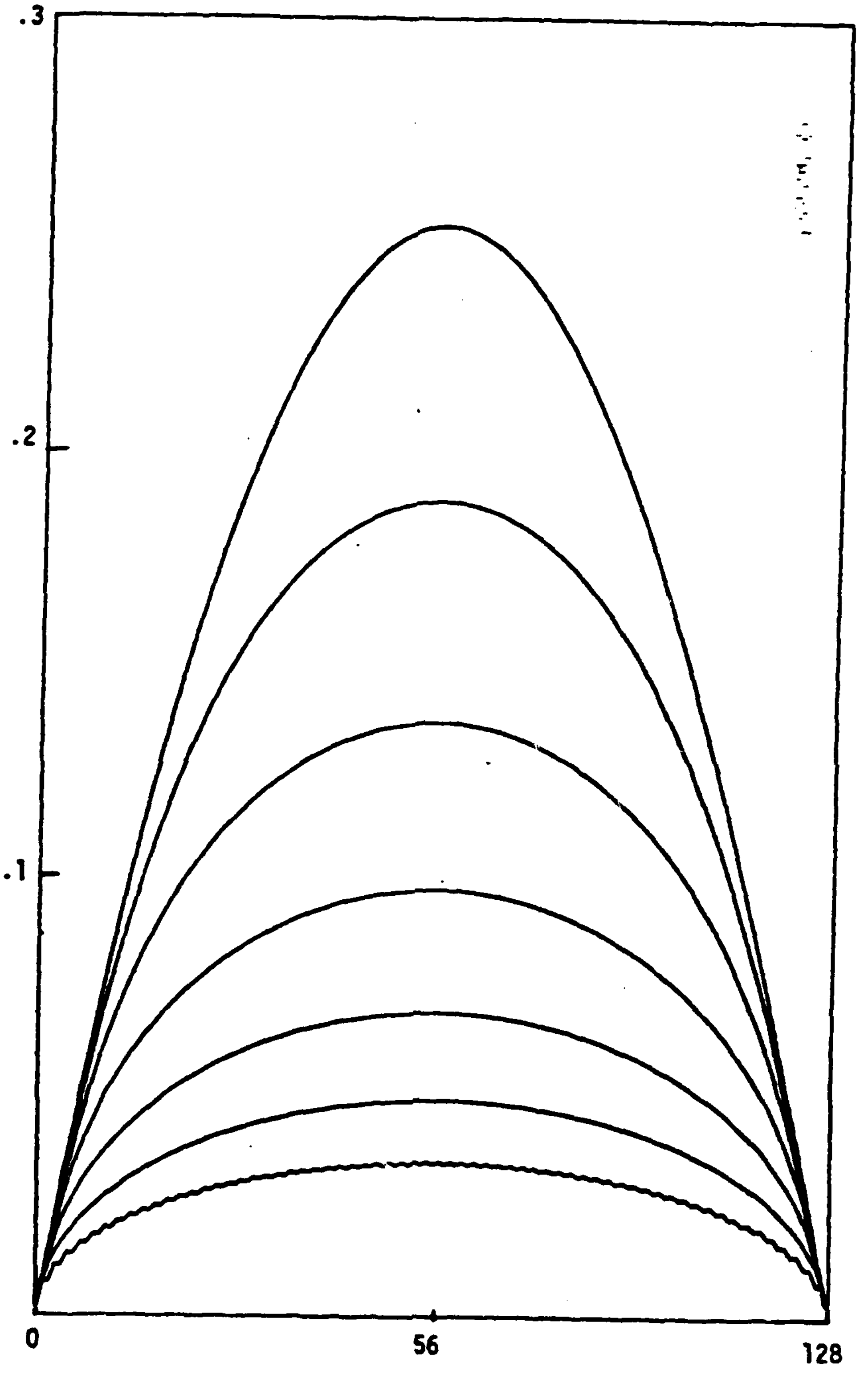


Figure 4: p vs. $f_k(p, 128)$, drawn as continuous curves. Curves are (from bottom to top) for $k = 1, 2, \ldots 7$.

the smaller tree, and Result 5 says the player is best off receiving his information just above the goal nodes.

Result 10: $f_{\log n}(2p,2n) \le f_{\log n}(p,n)$ with equality iff $p=0$ or $p=n$.

Proof: from Result 4,

$$\begin{aligned} f_{\log n}(2p,2n) &= \frac{2p\,(2n-2p)}{2n\,(2n-1)} \\ &= \frac{2p\,(n-p)}{n\,(2n-1)} \\ &< \frac{2p\,(n-p)}{n\,(2n-2)} \quad \text{if } 0<p<n \\ &= \frac{p\,(n-p)}{n\,(n-1)} \\ &= f_{\log n}(p,n) \quad \text{from Result 4 again.} \end{aligned}$$

If $p=0$ or $p=n$, $f_{\log n}(2p,2n)=0=f_{\log n}(p,n)$. ▪

Again we add a level to the search tree while maintaining the same proportion of goal nodes with value "1". This result says that in such a situation, behavior for the player who searches at the last possible moment is like that of the player who searches before his first move. In each case, the player expects to gain more from search in the smaller tree.

Here, however, the reasoning is different. When we explore near the bottom of the tree, the loss of search power in the larger tree occurs because of the increased "independence" of the goal nodes. In the smaller tree, if the left branch is not a goal node with value "1", that fact increases the probability that the right branch is a goal node with value "1" (because there are a fixed number of such goals). Hence if the left branch fails to give us the hoped-for "1", the right branch has better than its *a priori* chance of doing so. The same is true in the larger tree, but the increase in probability is less (the two nodes are closer to independent), so the gain from search is also less.

Conjecture 11: For $1 \le k \le \log n$,

$$f_k(2p,2n) \le f_k(p,n) \quad \text{with equality iff } p=0 \text{ or } p=n.$$

Results 9 and 10 established this statement for the two extremal strategies (searching before the first move and searching just before the last move). It seems likely that the statement holds for the in-between strategies as well. Computer results in Appendix B establish the truth of this conjecture for $n \le 128$.

Result 12: For any constant α between 0 and 1,

$$\lim_{n\to\infty} f_1(\alpha n,n) = 0$$

Proof: from formula 2 of Result 2,

$$f_1(\frac{n}{2}+1,n) = \frac{4}{n^2\binom{n}{n/2}} \left[\frac{n}{4}+1\right]\left[\frac{n}{4}\right]\binom{n/2+1}{n/4+1}\binom{n/2-1}{n/4}$$

$$= \frac{n+4}{4n}\,\frac{\binom{n/2+1}{n/4+1}\binom{n/2-1}{n/4}}{\binom{n}{n/2}}$$

$$\to 0 \quad \text{as } n\to\infty$$

Use Stirling's formula to demonstrate this convergence.

From Result 6, $f_1(p,n)$ is maximized when $p = \frac{n}{2}+1$. Result 12 follows by dominated convergence. ▪

In Result 12, we fix the proportion of "1"s at α and explore at the root of increasingly deep trees. The result says that for large enough trees a single exploration of the two subtrees of the root of the game tree prior to the player's first move adds almost nothing to his expected score. This is quite predictable: the player is too far from the goal nodes when he receives his information for it to avail him at all.

Conjecture 13: For any constant α between 0 and 1,

and for any fixed positive integer k,

$$\lim_{n \to \infty} f_k(\alpha n, n) = 0$$

This conjecture is a slight extension of Result 12 and should follow from Result 12 as a corollary. Since k is fixed, exploration takes place a long way from the goal nodes (just as in Result 12), so one would expect search to add almost nothing to the player's expected score.

Result 14: For any constant α between 0 and 1,

$$\lim_{n \to \infty} f_{\log n}(\alpha n, n) = \alpha(1-\alpha)$$

Proof: from Result 4,

$$\lim_{n \to \infty} f_{\log n}(\alpha n, n) = \lim_{n \to \infty} \frac{\alpha n(n - \alpha n)}{n(n-1)}$$

$$= \lim_{n \to \infty} \frac{\alpha(1-\alpha)n}{n-1}$$

$$= \alpha(1-\alpha) \bullet$$

This interesting result gives the convergence behavior for the player who searches only at the last possible moment. His expected gain from that search is the product of the fraction of goal nodes with value "1" and the fraction of goal nodes with value "0".

Suppose we were to assign the goal nodes value "1" or "0" independently with Pr("1") = α. We would expect Result 14 to approximate (and, in the limit, equal) behavior of last-minute search on such game trees. An easy calculation of the expected improvement by last-minute search on such trees does in fact yield the same answer $\alpha(1-\alpha)$ as given in Result 14.

4. Conclusions and Further Research

Given limited resources, what search strategy is best? We selected a model in which to answer this question, wherein we made precise such concepts as "search strategy" and "resources". No claim is made that the model is the only plausible one, nor that it accurately represents real-world games or decision-making. The model was selected for three reasons. First, it isolates that which we wish to study: uncertainty arising from incomplete search. Second, the model is simple enough to define the problem precisely and to allow hopes of obtaining analytic results. Third, the trees allowed and evaluation function used are sufficiently generic to model many interesting games played with many kinds of strategies.

The model features a division of "strategy" into "search strategy" and "move strategy"; resources specified as number of node evaluations allowed; node evaluations described as "exploration", wherein the player learns the probability mass function of possible outcomes were he to move at random beneath the evaluated node; and the meta-player assumption that all assignments of goal node values to goal nodes consistent with the given root blind search pmf are equiprobable.

We analyzed a class of strategies called one-time k strategies, on a class of trees called (p,n) trees. For these trees with these strategies, we proved the following results, among others.

1. A short formula, recursive in p, exists for $f_1(p,n)$. For $k \geq 2$, $f_k(p,n)$ can be expressed as a short sum, each of whose terms includes an f_1 term.

2. The longer the player postpones search, the better the strategy. That is, $f_k \leq f_{k+1}$, for $1 \leq k < \log n$. Hence the optimal strategy (among those allowed in the restricted case) is to explore at the leaves of the game tree. Further, this result [due in part to Bruce Ballard] applies even if we allow the goal nodes to have values other than "0" and "1".

3. The benefit from search is greatest in trees for which about half of the goal nodes have value "1" and half have value "0". More precisely, we can graph $f_k(p,n)$ as a continuous function of p. For $k \geq 2$ (and approximately for $k = 1$), $f_k(p,n)$ is a concave function symmetric about $p = \frac{n}{2}$.

4. In deep trees, search at the root gains nothing for the player. That is, the one-time 1 strategy is no better than random (no-search) play. Search just above the leaves (the one-time *log n* strategy) gains $\alpha(1-\alpha)$, where α is the fraction of goal nodes with value "1".

The girdle we wore herein to obtain analytic results can be loosened in two natural ways. First, we might consider trees with more than two possible goal values. For example, we might consider a (p,n) tree with one or two (in general, r) goal nodes allowed to have a third value. Second, we might consider more complicated strategies. For example, we might consider allowing two (in general, t) explorations, each at any level of the tree. Each of these extensions is under current investigation.

The long-term goal of this research is to compute the optimal search strategy in the unrestricted model. One hopes that progress toward that goal will help us better understand our intuitive idea of "search with limited resources".

5. Appendix A — Proofs

5.1. Proof of Result 1

Result 1: $f_k(p,n) = f_k(n-p,n)$.

Proof: Fix k and n throughout the proof.

Recall that in the one-time k strategy, the player moves randomly to level $k-1$ of the tree, explores the two subtrees beneath him, moves to the subtree with higher blind search expectation, and then moves randomly for the rest of the game. For any (m,n) tree played according to the one-time k strategy, define the following random variables.

A_m^l = fraction of "1"-valued goal nodes in the left subtree explored.

A_m^r = fraction of "1"-valued goal nodes in the right subtree explored.

B_m^l = fraction of "0"-valued goal nodes in the left subtree explored.

B_m^r = fraction of "0"-valued goal nodes in the right subtree explored.

These random variables are not independent of each other. However, because the player is using a one-time k strategy in an (m,n) tree, their distributions depend on only m, n and k. Since n and k are fixed, we omit noting the dependence on n and k.

Step 1: $(A_{n-p}^l, A_{n-p}^r) \approx (B_p^l, B_p^r)$, where $\approx$ stands for "has the same (joint) distribution as". This is true because when we interchange the "1"s and "0"s in the goal values of a $(n-p,n)$ tree, we obtain a (p,n) tree for which "1"s are interpreted as "0"s and vice-versa. A_{n-p}^l and A_{n-p}^r refer to the "1"s in a $(n-p,n)$ tree, while B_p^l and B_p^r refer to the "0"s in a (p,n) tree. Hence their joint distributions are identical.

Step 2: $A_p^l + B_p^l = 1$ (therefore $A_p^l = 1 - B_p^l$) since all goal nodes in the left subtree have value either "1" or "0". Similarly, $A_p^r = 1 - B_p^r$.

Step 3: $\mathrm{E}[\max\{A_p^l - \frac{p}{n}, A_p^r - \frac{p}{n}\}] = \mathrm{E}[\max\{\frac{p}{n} - A_p^l, \frac{p}{n} - A_p^r\}]$.

This symmetry result occurs because having one branch "bad" is (on average) exactly balanced by its brother being "good". More precisely:

$$\mathbf{E}\left[\max\left\{A_p^l - \frac{p}{n},\, A_p^r - \frac{p}{n}\right\}\right] - \mathbf{E}\left[\max\left\{\frac{p}{n} - A_p^l,\, \frac{p}{n} - A_p^r\right\}\right]$$

$$= \mathbf{E}\left[max\,\{A_p^l, A_p^r\}\right] - \frac{p}{n} - \left(\frac{p}{n} + \mathbf{E}\left[\max\{-A_p^l, -A_p^r\}\right]\right)$$

$$= \mathbf{E}\left[max\,\{A_p^l, A_p^r\}\right] + \mathbf{E}\left[\min\{A_p^l, A_p^r\}\right] - \frac{2p}{n}$$

since $\max(-X,-Y) = -\min(X,Y)$ for any random variables X and Y

$$= \mathbf{E}\left[max\,\{A_p^l, A_p^r\} + \min\{A_p^l, A_p^r\}\right] - \frac{2p}{n}$$

$$= \mathbf{E}\left[A_p^l + A_p^r\right] - \frac{2p}{n}$$

since $\max(X,Y) + \min(X,Y) = X+Y$ for any random variables X and Y

$$= \mathbf{E}[A_p^l] + \mathbf{E}[A_p^r] - \frac{2p}{n}$$

$$= \frac{p}{n} + \frac{p}{n} - \frac{2p}{n}$$ since we moved randomly to level $k-1$ of the tree

$$= 0$$

Step 4: The proof of Result 1 now proceeds as follows. By definition of f_k,

$f_k(n-p,n) = \mathbf{E}\left[\max\{A_{n-p}^l, A_{n-p}^r\}\right]$ − *blind search expectation of a* $(n-p,n)$ *tree*

$$= \mathbf{E}\left[\max\{B_p^l, B_p^r\}\right] - \frac{n-p}{n} \qquad \text{by Step 1}$$

$$= \mathbf{E}\left[\max\left\{\frac{p}{n} - (1-B_p^l),\, \frac{p}{n} - (1-B_p^r)\right\}\right]$$

$$= \mathrm{E}\left[\max\{\tfrac{p}{n} - A_p^l,\ \tfrac{p}{n} - A_p^r\}\right] \qquad \text{by Step 2}$$

$$= \mathrm{E}\left[\max\{A_p^l - \tfrac{p}{n},\ A_p^r - \tfrac{p}{n}\}\right] \qquad \text{by Step 3}$$

$$= \mathrm{E}\left[\max\{A_p^l, A_p^r\}\right] - \frac{p}{n}$$

$$= f_k(p,n) \qquad \text{by definition of } f_k$$

■

5.2. Proof of Result 2

Result 2:

$$f_1(p,n) = \frac{2}{n\binom{n}{n/2}} \sum_{j=\lfloor p/2+1\rfloor}^{n/2} [2j-p]\binom{p}{j}\binom{n-p}{n/2-j}$$

$$= \frac{4}{n^2\binom{n}{n/2}} \left\lfloor \frac{p}{2}+1 \right\rfloor \left\lfloor \frac{n-p+1}{2} \right\rfloor \binom{p}{\lfloor p/2+1\rfloor}\binom{n-p}{\lfloor (n-p+1)/2\rfloor}$$

$$= \begin{cases} 0 & \text{if } p=0 \text{ or } p=n \\ \dfrac{1}{n} & \text{if } p=1 \text{ or } p=n-1 \\ \dfrac{n-p}{n-p+1} f_1(p-1,n) & \text{if } 2 \le p \le n-2 \text{ and } p \text{ is } \textit{even} \\ \dfrac{p}{p-1} f_1(p-1,n) & \text{if } 3 \le p \le n-3 \text{ and } p \text{ is } \textit{odd} \end{cases}$$

$$= \begin{cases} 0 & \text{if } p=0 \text{ or } p=n \\ \dfrac{1}{n} \displaystyle\prod_{j=1}^{\lfloor p/2\rfloor} \frac{n-2j}{n-2j+1} \prod_{j=1}^{\lfloor (p-1)/2\rfloor} \frac{2j+1}{2j} & \text{if } 1 \le p \le n-1 \end{cases}$$

Proof: *(a) first formula*. In the one-time k strategy, the player examines the two subtrees of the root and moves to whichever subtree has more goal nodes with value "1". We divide the possible outcomes of the exploration into cases: case j means the selected subtree has j goal nodes with value "1". The improvement over blind search expectation in case j is ("b.s.e." stands for "blind search expectation"):

expected score in case j - *b.s.e. of a* (p,n) *tree*

$$= b.s.e.\ of\ a\ (j, \tfrac{n}{2})\ tree \;-\; b.s.e.\ of\ a\ (p,n)\ tree$$

$$= \frac{j}{n/2} - \frac{p}{n}$$

$$= \frac{1}{n}\,[2j-p] \qquad (*)$$

We need to compute the weight of each case. The number of ways to select the j "1"s in the chosen subtree is $\binom{p}{j}$. The number of ways to select the remaining "0"s is $\binom{n-p}{n/2-j}$. Further, the subtree selected could be either on the left or on the right (except for the $p/2$ case, to be handled shortly). The number of ways to select any subtree of the root is $\binom{n}{n/2}$. The weight of case j is

$$\frac{2\binom{p}{j}\binom{n-p}{n/2-j}}{\binom{n}{n/2}} \qquad (**)$$

To get the first formula in Result 1, we multiply the improvement (*) in each case by the weight (**) of each case, and sum over the cases. The $\frac{p}{2}$ case has zero improvement, so we may omit that case. The selected subtree must then hold more than half of the p goal nodes with value "1", so the sum begins at case $j = \left\lfloor \frac{p}{2}+1 \right\rfloor$. The sum ends when we have exhausted either all the goal nodes with value "1"

($j = p$) or all the goal nodes in the subtree ($j = \frac{n}{2}$). Since $\binom{p}{j}$ is interpreted as 0 if $j > p$, we can write the sum as if it continues until $j = \frac{n}{2}$.

(b) second formula. One can prove by induction on m the formula [KNUT 73]

$$\sum_{j=0}^{m} \binom{r}{j}\binom{s}{t-j}[tr-(r+s)j] = [m+1][t-m]\binom{r}{m+1}\binom{s}{t-m} \qquad (\dagger)$$

This formula holds for all integers r, s, t, and m.

We have from the first formula in Result 2 that

$$f_1(p,n) = \frac{2}{n\binom{n}{n/2}} \sum_{j=\lfloor p/2+1 \rfloor}^{n/2} [2j-p]\binom{p}{j}\binom{n-p}{n/2-j}$$

$$= \frac{2}{[-\frac{n}{2}]\,n\binom{n}{n/2}} \left[\sum_{j=0}^{n/2} [\tfrac{n}{2}p - nj]\binom{p}{j}\binom{n-p}{n/2-j} \right.$$

$$\left. - \sum_{j=0}^{\lfloor p/2 \rfloor} [\tfrac{n}{2}p - nj]\binom{p}{j}\binom{n-p}{n/2-j} \right]$$

$$= -\frac{4}{n^2\binom{n}{n/2}} \left[0 - \left\lfloor \frac{p}{2}+1 \right\rfloor \left\lceil \frac{n-p+1}{2} \right\rceil \binom{p}{\lfloor p/2+1 \rfloor}\binom{n-p}{\lfloor (n-p+1)/2 \rfloor} \right]$$

This last step follows by applying ($\dagger$) with $r=p$, $s=n-p$, $t=\frac{n}{2}$, and $m=\frac{n}{2}$ in the first summation and $m=\left\lfloor \frac{p}{2} \right\rfloor$ in the second summation.

(c) third formula. First suppose p is even, $2 \le p \le n-2$. Applying the second formula in Result 2, we get

$$f_1(p-1,n) = \frac{4}{n^2\binom{n}{n/2}} \left\lfloor \frac{p-1}{2}+1 \right\rfloor \left\lceil \frac{n-p+2}{2} \right\rceil \binom{p-1}{\lfloor (p-1)/2+1 \rfloor}\binom{n-p+1}{\lfloor (n-p+2)/2 \rfloor}$$

$$= \frac{4}{n^2\binom{n}{n/2}} \left[\frac{p}{2}\right] \left[\frac{n-p+2}{2}\right] \binom{p-1}{p/2} \binom{n-p+1}{[n-p+2]/2}$$

$$= \frac{4}{n^2\binom{n}{n/2}} \left[\frac{p}{2}\right] \left[\frac{n-p+2}{2}\right] \binom{p}{p/2+1} \left[\frac{\frac{p}{2}+1}{p}\right] \binom{n-p}{[n-p]/2} \left[\frac{n-p+1}{\frac{n-p+2}{2}}\right]$$

$$= \frac{4}{n^2\binom{n}{n/2}} \left[\frac{p}{2}+1\right] \left[\frac{n-p}{2}\right] \binom{p}{p/2+1} \binom{n-p}{[n-p]/2} \left[\frac{n-p+1}{n-p}\right]$$

$$= f_1(p,n)\,\frac{n-p+1}{n-p} \qquad \text{since } p \text{ is even}$$

A similar application of the second formula in Result 2 yields the appropriate formula for odd p. The $p = 0,\ 1,\ n-1,$ and n cases also follow immediately by applying the second formula in Result 2.

(d) fourth formula. This follows directly from repeated applications of the third formula in Result 2. ▪

5.3. Proof of Result 5

Result 5 [Ballard and Hutchler]: for $1 \leq k < \log n$,

$$f_k(p,n) \leq f_{k+1}(p,n)$$

with equality iff $p=0$ or $p=1$ or $p=n-1$ or $p=n$.

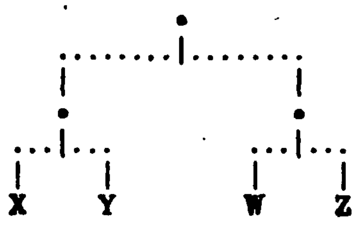
Further, the proof does not depend on the goal node values being restricted to "0" and "1", and works for any complete n-ary tree.

Proof: Key ideas in this proof were supplied by Bruce Ballard after a conversation with the author.

Result 4 establishes that equality holds if $p = 0, 1, n-1$, or n. Let $2 \le p \le n-2$.

Step 1: Show that $f_1(p,n) < f_2(p,n)$.

Let the random variables X, Y, W and Z be the number of goal nodes with value "1" in the four subtrees having root at level 2 of the (p,n) tree, respectively.



Then by definition,

$$f_1(p,n) = \mathrm{E}\left[\max\left\{\frac{X+Y}{n/2}, \frac{W+Z}{n/2}\right\} - \frac{p}{n}\right]$$

$$= \frac{2}{n}\mathrm{E}[\max\{X+Y, W+Z\}] - \frac{p}{n}$$

Since the one-time 2 strategist makes his first move at random,

$$f_2(p,n) = \frac{1}{2}\mathrm{E}\left[\max\left\{\frac{X}{n/4}, \frac{Y}{n/4}\right\} - \frac{p}{n}\right] + \frac{1}{2}\mathrm{E}\left[\max\left\{\frac{W}{n/4}, \frac{Z}{n/4}\right\} - \frac{p}{n}\right]$$

$$= \frac{2}{n}\Big[\mathrm{E}[\max\{X,Y\}] + \mathrm{E}[\max\{W,Z\}]\Big] - \frac{p}{n}$$

Hence to show $f_1(p,n) < f_2(p,n)$ it will suffice to show that

$$\mathrm{E}[\max\{X+Y, W+Z\}] < \mathrm{E}[\max\{X,Y\}] + \mathrm{E}[\max\{W,Z\}]$$

Now $(X,Y) \approx (X,W)$ and $(W,Z) \approx (Y,Z)$, where $\approx$ stands for "has the same joint distribution", so $max(X,Y) \approx max(X,W)$ and $max(W,Z) \approx max(Y,Z)$. Taking expectations, we get

$$\mathrm{E}[\max\{X,Y\}] = \mathrm{E}[\max\{X,W\}] \quad \text{and} \quad \mathrm{E}[\max\{W,Z\}] = \mathrm{E}[\max\{Y,Z\}] \qquad (*)$$

For any real numbers (hence any random variables),

$$\max\{X+Y\,,\,W+Z\} \le \max\{X,W\} + \max\{Y,Z\} \quad \text{so}$$

$$\mathrm{E}[\max\{X+Y\,,\,W+Z\}] \le \mathrm{E}[\max\{X,W\} + \max\{Y,Z\}]$$

Further, this inequality is strict as long as Pr $(X>W$ and $Y<Z)$ is non-zero. Since $2 \le p \le n-2$, such is the case here. We have

$$\begin{aligned}\mathrm{E}[\max\{X+Y\,,\,W+Z\}] &< \mathrm{E}[\max\{X,W\} + \max\{Y,Z\}] \\ &= \mathrm{E}[\max\{X,W\}] + \mathrm{E}[\max\{Y,Z\}] \\ &= \mathrm{E}[\max\{X,Y\}] + \mathrm{E}[\max\{W,Z\}] \quad \text{by } (*) \text{ above}\end{aligned}$$

As noted earlier, this suffices to conclude Step 1 of the proof.

Step 2: Show that $f_k(p,n) < f_{k+1}(p,n)$ if $2 \le k < \log n$.

$f_k(p,n) = f_1$ applied to a random subtree with root at level $k-1$

$\le f_2$ applied to a random subtree with root at level $k-1$

(because $f_1 \le f_2$ on *any* tree)

$= f_{k+1}(p,n)$.

Further, the inequality is strict because there is a non-zero probability that the random subtree with root at level $k-1$ has between two and all but two goal nodes with value "1". ▪

5.4. Proof of Result 6

Result 6:

$$f_1(p,n) > f_1(p-2,n) \quad \text{if } 2 \le p \le \frac{n}{2}$$

$$f_1(p,n) > f_1(p+2,n) \quad \text{if } \frac{n}{2} \le p \le n-2$$

$$f_1(p,n) < f_1(p-1,n) \quad \text{if } p \text{ is even}$$

$$f_1(p,n) > f_1(p-1,n) \quad \text{if } p \text{ is odd}$$

In particular, for fixed n, $f_1(p,n)$ is largest when $p = \frac{n}{2}-1$ or $p = \frac{n}{2}+1$.

Proof: (a) Suppose $2 \le p \le \frac{n}{2}$. Then $p \le n-p$ so $p-2 < n-p$ and $p-1 < n-p+1$. It follows that

$$\frac{p-2}{p-1} < \frac{n-p}{n-p+1} \quad \text{and} \qquad (*)$$

$$\frac{p-1}{p} < \frac{n-p+1}{n-p+2} \qquad (**)$$

For $p = 2$, the result is trivial. For larger even p, Result 2 shows that

$$f_1(p,n) = \frac{n-p}{n-p+1} f_1(p-1,n)$$

$$= \frac{n-p}{n-p+1}\,\frac{p-1}{p-2} f_1(p-2,n) \quad \text{using Result 2 again, on odd } p-1$$

But from (*), we have $\frac{n-p}{n-p+1}\,\frac{p-1}{p-2} > 1$, so $f_1(p,n) > f_1(p-2,n)$ for even p. For odd p, Result 2 shows that

$$f_1(p,n) = \frac{p}{p-1} f_1(p-1,n)$$

$$= \frac{p}{p-1}\,\frac{n-p+1}{n-p+2}\,f_1(p-2,n) \quad \text{using Result 2 again, on even } p-1$$

Since (**) implies that $\frac{p}{p-1}\,\frac{n-p+1}{n-p+2} > 1$, we have that $f_1(p,n) > f_1(p-2,n)$ for odd p.

(b) If $\frac{n}{2} \le p \le n-2$, part (a) of this proof and the symmetry condition of Result 1 entail that $f_1(p,n) > f_1(p+2,n)$.

(c) If p is even, from Result 2

$$f_1(p,n) = \frac{n-p}{n-p+1}\,f_1(p-1,n)$$

$$< f_1(p-1,n) \qquad \text{since } \frac{n-p}{n-p+1} < 1$$

(d) If p is odd, from Result 2 we have

$$f_1(p,n) = \frac{p}{p-1}\,f_1(p-1,n)$$

$$> f_1(p-1,n) \qquad \text{since } \frac{p}{p-1} > 1$$

■

5.5. Proof of Result 9

Result 9: $f_1(2p,2n) \le f_1(p,n)$ with equality iff $p=0$ or $p=n$.

Proof: If $p=0$ or $p=n$, the result follows from Result 4. Let $0 < p < n$.

Step 1: Show true for even p.

Basis case ($p = 2$):

$$f_1(4,2n) = \frac{2n-4}{2n-3}\,\frac{3}{2}\,\frac{2n-2}{2n-1}\,\frac{1}{2n} \quad \text{by repeated application of Result 2}$$

$$= \frac{n-2}{2(n-\frac{3}{2})} \frac{3(n-1)}{2(n-\frac{1}{2})} \frac{1}{n}$$

$$< \frac{3(n-1)}{4(n-\frac{3}{2})} \frac{n-2}{n-1} \frac{1}{n} \qquad \text{since } \frac{1}{n-\frac{1}{2}} < \frac{1}{n-1}$$

$$\le \frac{n-2}{n-1} \frac{1}{n} \qquad \text{if } n \ge 3 \text{ (true here)}$$

$$= f_1(2,n) \qquad \text{by Result 2}$$

Induction step: Assume true for $p-2$ (p even, $p \ge 4$). Show true for p.

$$f_1(2p,2n) = \frac{2n-2p}{2n-2p+1} \frac{2p-1}{2p-2} \frac{2n-2p+2}{2n-2p+3} \frac{2p-3}{2p-4} f_1(2p-4,2n)$$

by repeated application of Result 2

$$< \frac{2n-2p}{2n-2p+1} \frac{2p-1}{2p-2} \frac{2n-2p+2}{2n-2p+3} \frac{2p-3}{2p-4} f_1(p-2,n)$$

by the induction hypothesis

$$= \frac{2n-2p}{2n-2p+1} \frac{2p-1}{2p-2} \frac{2n-2p+2}{2n-2p+3} \frac{2p-3}{2p-4} \frac{p-2}{p-1} \frac{n-p+1}{n-p} f_1(p,n) \qquad (*)$$

by repeated application of Result 2

But calculations show that the coefficient of $f_1(p,n)$ in the line marked (*) is less than or equal to 1 precisely when $p \le \frac{n}{2}+1$. Hence the induction step follows for $p \le \frac{n}{2}$. For larger even p, the result follows from the symmetry condition in Result 1.

Step 2: Show true for odd p.

Basis case ($p = 1$):

$$f_1(2,2n) = \frac{2n-2}{2n-1}\,\frac{1}{2n} \qquad \text{by two applications of Result 2}$$

$$= \frac{n-1}{2n-1}\,\frac{1}{n}$$

$$< \frac{1}{n}$$

$$= f_1(1,n)$$

Induction step: We will show true for odd $p > 1$, using Step 1 as the "induction hypothesis". Let $p > 1$ be odd.

$$f_1(2p,2n) = \frac{2n-2p}{2n-2p+1}\,\frac{2p-1}{2p-2}\,f_1(2p-2,2n) \qquad \text{by Result 2}$$

$$< \frac{2n-2p}{2n-2p+1}\,\frac{2p-1}{2p-2}\,f_1(p-1,n) \qquad \text{using Step 1, since } 2p-2 \text{ is even}$$

$$= \frac{2n-2p}{2n-2p+1}\,\frac{2p-1}{2p-2}\,\frac{p-1}{p}\,f_1(p,n) \qquad \text{by Result 2}$$

$$= \frac{n-p}{n-p+\frac{1}{2}}\,\frac{p-\frac{1}{2}}{p}\,f_1(p,n)$$

$$< f_1(p,n)$$

●

6. Appendix B --- Computer Results

```
Depth of 0-1 tree    = 1

Number of goal nodes = 2


Number  Improvement from learning  about level
of 1's     1
------     -

   0    0.0000
   1    0.5000
   2    0.0000
```

```
Depth of 0-1 tree    = 2

Number of goal nodes = 4


Number  Improvement from learning  about level
of 1's     1       2
------     -       -

   0    0.0000  0.0000
   1    0.2500  0.2500
   2    0.1667  0.3333
   3    0.2500  0.2500
   4    0.0000  0.0000
```

```
Depth of 0-1 tree     = 3

Number of goal nodes = 8

Number   Improvement from learning  about level
of 1's       1         2         3
------       -         -         -

   0     0.0000   0.0000   0.0000
   1     0.1250   0.1250   0.1250
   2     0.1071   0.1786   0.2143
   3     0.1607   0.1964   0.2679
   4     0.1286   0.2000   0.2857
   5     0.1607   0.1964   0.2679
   6     0.1071   0.1786   0.2143
   7     0.1250   0.1250   0.1250
   8     0.0000   0.0000   0.0000
```

Depth of 0-1 tree = 4

Number of goal nodes = 16

Number of 1's	Improvement from learning about level 1	2	3	4
0	0.0000	0.0000	0.0000	0.0000
1	0.0625	0.0625	0.0625	0.0625
2	0.0583	0.0917	0.1083	0.1167
3	0.0875	0.1089	0.1411	0.1625
4	0.0808	0.1214	0.1637	0.2000
5	0.1010	0.1307	0.1788	0.2292
6	0.0918	0.1370	0.1882	0.2500
7	0.1071	0.1405	0.1933	0.2625
8	0.0952	0.1416	0.1949	0.2667
9	0.1071	0.1405	0.1933	0.2625
10	0.0918	0.1370	0.1882	0.2500
11	0.1010	0.1307	0.1788	0.2292
12	0.0808	0.1214	0.1637	0.2000
13	0.0875	0.1089	0.1411	0.1625
14	0.0583	0.0917	0.1083	0.1167
15	0.0625	0.0625	0.0625	0.0625
16	0.0000	0.0000	0.0000	0.0000

Depth of 0-1 tree = 5

Number of goal nodes = 32

Number of 1's	Improvement from learning about level 1	2	3	4	5
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0313	0.0313	0.0313	0.0313	0.0313
2	0.0302	0.0464	0.0544	0.0585	0.0605
3	0.0454	0.0567	0.0720	0.0821	0.0877
4	0.0438	0.0648	0.0856	0.1024	0.1129
5	0.0547	0.0716	0.0964	0.1198	0.1361
6	0.0527	0.0772	0.1052	0.1347	0.1573
7	0.0615	0.0820	0.1125	0.1472	0.1764
8	0.0590	0.0860	0.1186	0.1577	0.1935
9	0.0664	0.0895	0.1237	0.1665	0.2087
10	0.0635	0.0923	0.1280	0.1736	0.2218
11	0.0699	0.0947	0.1315	0.1793	0.2329
12	0.0666	0.0966	0.1343	0.1838	0.2419
13	0.0721	0.0980	0.1365	0.1872	0.2490
14	0.0683	0.0990	0.1380	0.1895	0.2540
15	0.0732	0.0996	0.1389	0.1909	0.2571
16	0.0689	0.0998	0.1392	0.1913	0.2581
17	0.0732	0.0996	0.1389	0.1909	0.2571
18	0.0683	0.0990	0.1380	0.1895	0.2540
19	0.0721	0.0980	0.1365	0.1872	0.2490
20	0.0666	0.0966	0.1343	0.1838	0.2419
21	0.0699	0.0947	0.1315	0.1793	0.2329
22	0.0635	0.0923	0.1280	0.1736	0.2218
23	0.0664	0.0895	0.1237	0.1665	0.2087
24	0.0590	0.0860	0.1186	0.1577	0.1935
25	0.0615	0.0820	0.1125	0.1472	0.1764
26	0.0527	0.0772	0.1052	0.1347	0.1573
27	0.0547	0.0716	0.0964	0.1198	0.1361
28	0.0438	0.0648	0.0856	0.1024	0.1129
29	0.0454	0.0567	0.0720	0.0821	0.0877
30	0.0302	0.0464	0.0544	0.0585	0.0605
31	0.0313	0.0313	0.0313	0.0313	0.0313
32	0.0000	0.0000	0.0000	0.0000	0.0000

Depth of 0-1 tree = 6

Number of goal nodes = 64

Number of 1's	Improvement from learning about level 1	2	3	4	5	6
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0156	0.0156	0.0156	0.0156	0.0156	0.0156
2	0.0154	0.0233	0.0273	0.0293	0.0303	0.0308
3	0.0231	0.0288	0.0363	0.0412	0.0439	0.0454
4	0.0227	0.0333	0.0436	0.0517	0.0567	0.0595
5	0.0284	0.0372	0.0496	0.0610	0.0687	0.0732
6	0.0279	0.0405	0.0548	0.0692	0.0798	0.0863
7	0.0325	0.0435	0.0593	0.0765	0.0902	0.0990
8	0.0320	0.0462	0.0634	0.0830	0.0998	0.1111
9	0.0359	0.0487	0.0670	0.0888	0.1087	0.1228
10	0.0353	0.0509	0.0703	0.0941	0.1170	0.1339
11	0.0388	0.0529	0.0733	0.0989	0.1247	0.1446
12	0.0381	0.0548	0.0761	0.1032	0.1318	0.1548
13	0.0413	0.0565	0.0786	0.1071	0.1383	0.1644
14	0.0405	0.0581	0.0809	0.1107	0.1444	0.1736
15	0.0433	0.0596	0.0831	0.1140	0.1499	0.1823
16	0.0425	0.0609	0.0851	0.1170	0.1550	0.1905
17	0.0451	0.0622	0.0869	0.1197	0.1596	0.1982
18	0.0442	0.0633	0.0885	0.1222	0.1638	0.2054
19	0.0466	0.0644	0.0900	0.1245	0.1676	0.2121
20	0.0456	0.0653	0.0914	0.1266	0.1711	0.2183
21	0.0479	0.0662	0.0927	0.1285	0.1742	0.2240
22	0.0467	0.0670	0.0938	0.1302	0.1770	0.2292
23	0.0489	0.0676	0.0948	0.1317	0.1795	0.2339
24	0.0477	0.0683	0.0957	0.1330	0.1816	0.2381
25	0.0497	0.0688	0.0965	0.1342	0.1835	0.2418
26	0.0484	0.0693	0.0972	0.1352	0.1851	0.2450
27	0.0502	0.0697	0.0977	0.1360	0.1864	0.2478
28	0.0489	0.0700	0.0982	0.1367	0.1875	0.2500
29	0.0506	0.0702	0.0986	0.1373	0.1884	0.2517
30	0.0492	0.0704	0.0988	0.1376	0.1890	0.2530
31	0.0508	0.0705	0.0990	0.1379	0.1893	0.2537
32	0.0493	0.0705	0.0990	0.1379	0.1894	0.2540
33	0.0508	0.0705	0.0990	0.1379	0.1893	0.2537
34	0.0492	0.0704	0.0988	0.1376	0.1890	0.2530
35	0.0506	0.0702	0.0986	0.1373	0.1884	0.2517
36	0.0489	0.0700	0.0982	0.1367	0.1875	0.2500
37	0.0502	0.0697	0.0977	0.1360	0.1864	0.2478
38	0.0484	0.0693	0.0972	0.1352	0.1851	0.2450
39	0.0497	0.0688	0.0965	0.1342	0.1835	0.2418
40	0.0477	0.0683	0.0957	0.1330	0.1816	0.2381
41	0.0489	0.0676	0.0948	0.1317	0.1795	0.2339
42	0.0467	0.0670	0.0938	0.1302	0.1770	0.2292
43	0.0479	0.0662	0.0927	0.1285	0.1742	0.2240
44	0.0456	0.0653	0.0914	0.1266	0.1711	0.2183

45	0.0466	0.0644	0.0900	0.1245	0.1676	0.2121
46	0.0442	0.0633	0.0885	0.1222	0.1638	0.2054
47	0.0451	0.0622	0.0869	0.1197	0.1596	0.1982
48	0.0425	0.0609	0.0851	0.1170	0.1550	0.1905
49	0.0433	0.0596	0.0831	0.1140	0.1499	0.1823
50	0.0405	0.0581	0.0809	0.1107	0.1444	0.1736
51	0.0413	0.0565	0.0786	0.1071	0.1383	0.1644
52	0.0381	0.0548	0.0761	0.1032	0.1318	0.1548
53	0.0388	0.0529	0.0733	0.0989	0.1247	0.1446
54	0.0353	0.0509	0.0703	0.0941	0.1170	0.1339
55	0.0359	0.0487	0.0670	0.0888	0.1087	0.1228
56	0.0320	0.0462	0.0634	0.0830	0.0998	0.1111
57	0.0325	0.0435	0.0593	0.0765	0.0902	0.0990
58	0.0279	0.0405	0.0548	0.0692	0.0798	0.0863
59	0.0284	0.0372	0.0496	0.0610	0.0687	0.0732
60	0.0227	0.0333	0.0436	0.0517	0.0567	0.0595
61	0.0231	0.0288	0.0363	0.0412	0.0439	0.0454
62	0.0154	0.0233	0.0273	0.0293	0.0303	0.0308
63	0.0156	0.0156	0.0156	0.0156	0.0156	0.0156
64	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Depth of 0-1 tree = 7

Number of goal nodes = 128

Number	Improvement from learning about level						
of 1's	1	2	3	4	5	6	7
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0078	0.0078	0.0078	0.0078	0.0078	0.0078	0.0078
2	0.0078	0.0117	0.0137	0.0146	0.0151	0.0154	0.0155
3	0.0116	0.0145	0.0182	0.0206	0.0220	0.0227	0.0231
4	0.0115	0.0169	0.0220	0.0260	0.0284	0.0298	0.0305
5	0.0144	0.0189	0.0251	0.0307	0.0345	0.0367	0.0378
6	0.0143	0.0207	0.0279	0.0350	0.0402	0.0433	0.0450
7	0.0167	0.0223	0.0304	0.0388	0.0455	0.0497	0.0521
8	0.0165	0.0238	0.0326	0.0424	0.0505	0.0559	0.0591
9	0.0186	0.0252	0.0347	0.0456	0.0553	0.0619	0.0659
10	0.0185	0.0265	0.0366	0.0486	0.0597	0.0677	0.0726
11	0.0203	0.0277	0.0383	0.0513	0.0639	0.0733	0.0792
12	0.0201	0.0289	0.0400	0.0539	0.0679	0.0788	0.0856
13	0.0218	0.0299	0.0415	0.0563	0.0717	0.0840	0.0920
14	0.0216	0.0309	0.0430	0.0585	0.0752	0.0890	0.0982
15	0.0232	0.0319	0.0444	0.0606	0.0786	0.0939	0.1043
16	0.0230	0.0328	0.0457	0.0626	0.0818	0.0986	0.1102
17	0.0244	0.0337	0.0470	0.0645	0.0848	0.1031	0.1161
18	0.0242	0.0345	0.0482	0.0663	0.0877	0.1075	0.1218
19	0.0255	0.0353	0.0494	0.0680	0.0904	0.1117	0.1274
20	0.0253	0.0361	0.0505	0.0697	0.0930	0.1157	0.1329
21	0.0265	0.0368	0.0515	0.0712	0.0955	0.1196	0.1382
22	0.0263	0.0375	0.0525	0.0727	0.0978	0.1234	0.1435
23	0.0275	0.0382	0.0535	0.0741	0.1001	0.1270	0.1486
24	0.0272	0.0388	0.0544	0.0755	0.1022	0.1304	0.1535
25	0.0284	0.0395	0.0553	0.0768	0.1042	0.1338	0.1584
26	0.0281	0.0400	0.0562	0.0781	0.1062	0.1370	0.1631
27	0.0292	0.0406	0.0570	0.0793	0.1081	0.1401	0.1678
28	0.0289	0.0412	0.0578	0.0804	0.1098	0.1430	0.1722
29	0.0299	0.0417	0.0585	0.0815	0.1115	0.1458	0.1766
30	0.0296	0.0422	0.0592	0.0826	0.1132	0.1486	0.1809
31	0.0306	0.0427	0.0599	0.0836	0.1147	0.1512	0.1850
32	0.0303	0.0431	0.0606	0.0846	0.1162	0.1536	0.1890
33	0.0312	0.0436	0.0612	0.0855	0.1176	0.1560	0.1929
34	0.0309	0.0440	0.0619	0.0864	0.1190	0.1583	0.1966
35	0.0318	0.0444	0.0624	0.0872	0.1203	0.1605	0.2002
36	0.0315	0.0448	0.0630	0.0881	0.1215	0.1625	0.2037
37	0.0323	0.0452	0.0635	0.0888	0.1227	0.1645	0.2071
38	0.0320	0.0455	0.0640	0.0896	0.1238	0.1664	0.2104
39	0.0328	0.0459	0.0645	0.0903	0.1249	0.1682	0.2135
40	0.0325	0.0462	0.0650	0.0910	0.1259	0.1699	0.2165
41	0.0333	0.0465	0.0654	0.0916	0.1269	0.1715	0.2194
42	0.0329	0.0468	0.0659	0.0922	0.1278	0.1730	0.2222
43	0.0337	0.0471	0.0663	0.0928	0.1287	0.1745	0.2248
44	0.0333	0.0474	0.0667	0.0934	0.1295	0.1758	0.2274

45	0.0340	0.0476	0.0670	0.0939	0.1303	0.1771	0.2298
46	0.0336	0.0478	0.0674	0.0944	0.1310	0.1783	0.2320
47	0.0343	0.0481	0.0677	0.0949	0.1317	0.1795	0.2342
48	0.0339	0.0483	0.0680	0.0953	0.1324	0.1805	0.2362
49	0.0346	0.0485	0.0683	0.0957	0.1330	0.1815	0.2381
50	0.0342	0.0487	0.0685	0.0961	0.1336	0.1824	0.2399
51	0.0349	0.0488	0.0688	0.0964	0.1341	0.1833	0.2416
52	0.0344	0.0490	0.0690	0.0967	0.1346	0.1841	0.2431
53	0.0351	0.0491	0.0692	0.0970	0.1350	0.1848	0.2445
54	0.0346	0.0493	0.0694	0.0973	0.1354	0.1854	0.2458
55	0.0353	0.0494	0.0695	0.0976	0.1358	0.1860	0.2470
56	0.0348	0.0495	0.0697	0.0978	0.1361	0.1865	0.2480
57	0.0354	0.0496	0.0698	0.0980	0.1364	0.1870	0.2490
58	0.0349	0.0496	0.0699	0.0981	0.1366	0.1874	0.2498
59	0.0355	0.0497	0.0700	0.0983	0.1369	0.1877	0.2504
60	0.0350	0.0498	0.0701	0.0984	0.1370	0.1880	0.2510
61	0.0356	0.0498	0.0702	0.0985	0.1372	0.1882	0.2514
62	0.0350	0.0498	0.0702	0.0985	0.1373	0.1884	0.2517
63	0.0356	0.0499	0.0702	0.0986	0.1373	0.1884	0.2519
64	0.0351	0.0499	0.0703	0.0986	0.1373	0.1885	0.2520
65	0.0356	0.0499	0.0702	0.0986	0.1373	0.1884	0.2519
66	0.0350	0.0498	0.0702	0.0985	0.1373	0.1884	0.2517
67	0.0356	0.0498	0.0702	0.0985	0.1372	0.1882	0.2514
68	0.0350	0.0498	0.0701	0.0984	0.1370	0.1880	0.2510
69	0.0355	0.0497	0.0700	0.0983	0.1369	0.1877	0.2504
70	0.0349	0.0496	0.0699	0.0981	0.1366	0.1874	0.2498
71	0.0354	0.0496	0.0698	0.0980	0.1364	0.1870	0.2490
72	0.0348	0.0495	0.0697	0.0978	0.1361	0.1865	0.2480
73	0.0353	0.0494	0.0695	0.0976	0.1358	0.1860	0.2470
74	0.0346	0.0493	0.0694	0.0973	0.1354	0.1854	0.2458
75	0.0351	0.0491	0.0692	0.0970	0.1350	0.1848	0.2445
76	0.0344	0.0490	0.0690	0.0967	0.1346	0.1841	0.2431
77	0.0349	0.0488	0.0688	0.0964	0.1341	0.1833	0.2416
78	0.0342	0.0487	0.0685	0.0961	0.1336	0.1824	0.2399
79	0.0346	0.0485	0.0683	0.0957	0.1330	0.1815	0.2381
80	0.0339	0.0483	0.0680	0.0953	0.1324	0.1805	0.2362
81	0.0343	0.0481	0.0677	0.0949	0.1317	0.1795	0.2342
82	0.0336	0.0478	0.0674	0.0944	0.1310	0.1783	0.2320
83	0.0340	0.0476	0.0670	0.0939	0.1303	0.1771	0.2298
84	0.0333	0.0474	0.0667	0.0934	0.1295	0.1758	0.2274
85	0.0337	0.0471	0.0663	0.0928	0.1287	0.1745	0.2248
86	0.0329	0.0468	0.0659	0.0922	0.1278	0.1730	0.2222
87	0.0333	0.0465	0.0654	0.0916	0.1269	0.1715	0.2194
88	0.0325	0.0462	0.0650	0.0910	0.1259	0.1699	0.2165
89	0.0328	0.0459	0.0645	0.0903	0.1249	0.1682	0.2135
90	0.0320	0.0455	0.0640	0.0896	0.1238	0.1664	0.2104
91	0.0323	0.0452	0.0635	0.0888	0.1227	0.1645	0.2071
92	0.0315	0.0448	0.0630	0.0881	0.1215	0.1625	0.2037
93	0.0318	0.0444	0.0624	0.0872	0.1203	0.1605	0.2002
94	0.0309	0.0440	0.0619	0.0864	0.1190	0.1583	0.1966
95	0.0312	0.0436	0.0612	0.0855	0.1176	0.1560	0.1929
96	0.0303	0.0431	0.0606	0.0846	0.1162	0.1536	0.1890
97	0.0306	0.0427	0.0599	0.0836	0.1147	0.1512	0.1850
98	0.0296	0.0422	0.0592	0.0826	0.1132	0.1486	0.1809

99	0.0299	0.0417	0.0585	0.0815	0.1115	0.1458	0.1766
100	0.0289	0.0412	0.0578	0.0804	0.1098	0.1430	0.1722
101	0.0292	0.0406	0.0570	0.0793	0.1081	0.1401	0.1678
102	0.0281	0.0400	0.0562	0.0781	0.1062	0.1370	0.1631
103	0.0284	0.0395	0.0553	0.0768	0.1042	0.1338	0.1584
104	0.0272	0.0388	0.0544	0.0755	0.1022	0.1304	0.1535
105	0.0275	0.0382	0.0535	0.0741	0.1001	0.1270	0.1486
106	0.0263	0.0375	0.0525	0.0727	0.0978	0.1234	0.1435
107	0.0265	0.0368	0.0515	0.0712	0.0955	0.1196	0.1382
108	0.0253	0.0361	0.0505	0.0697	0.0930	0.1157	0.1329
109	0.0255	0.0353	0.0494	0.0680	0.0904	0.1117	0.1274
110	0.0242	0.0345	0.0482	0.0663	0.0877	0.1075	0.1218
111	0.0244	0.0337	0.0470	0.0645	0.0848	0.1031	0.1161
112	0.0230	0.0328	0.0457	0.0626	0.0818	0.0986	0.1102
113	0.0232	0.0319	0.0444	0.0606	0.0786	0.0939	0.1043
114	0.0216	0.0309	0.0430	0.0585	0.0752	0.0890	0.0982
115	0.0218	0.0299	0.0415	0.0563	0.0717	0.0840	0.0920
116	0.0201	0.0289	0.0400	0.0539	0.0679	0.0788	0.0856
117	0.0203	0.0277	0.0383	0.0513	0.0639	0.0733	0.0792
118	0.0185	0.0265	0.0366	0.0486	0.0597	0.0677	0.0726
119	0.0186	0.0252	0.0347	0.0456	0.0553	0.0619	0.0659
120	0.0165	0.0238	0.0326	0.0424	0.0505	0.0559	0.0591
121	0.0167	0.0223	0.0304	0.0388	0.0455	0.0497	0.0521
122	0.0143	0.0207	0.0279	0.0350	0.0402	0.0433	0.0450
123	0.0144	0.0189	0.0251	0.0307	0.0345	0.0367	0.0378
124	0.0115	0.0169	0.0220	0.0260	0.0284	0.0298	0.0305
125	0.0116	0.0145	0.0182	0.0206	0.0220	0.0227	0.0231
126	0.0078	0.0117	0.0137	0.0146	0.0151	0.0154	0.0155
127	0.0078	0.0078	0.0078	0.0078	0.0078	0.0078	0.0078
128	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

REFERENCES

[BIER 78] Biermann, A. W.

Theoretical issues related to computer game playing programs.

Personal Computing,

September 1978.

[KNUT 79] Knuth, Donald E.

The Art of Computer Programming,

Vol. 1, Addison-Wesley, Reading, Mass., 1973, p. 72.

[NILS 80] Nilsson, Nils J.

Principles of Artificial Intelligence,

Tioga Publishing Co., Palo Alto, Calif., 1980, p. 6.

[TRUS 79] Truscott, Tom.

Minimum variance tree searching.

First International Symposium on Policy Analysis and Information Systems,

Duke University, Durham, N.C., June 1979.

[TRUS 82] Truscott, Tom.

Personal communication, 1982.